Support Constraint Machines

Marco Gori and Stefano Melacci

Department of Information Engineering
University of Siena, 53100 Siena, Italy
{marco,mela}@dii.unisi.it

Abstract. The significant evolution of kernel machines in the last few years has opened the doors to a truly new wave in machine learning on both the theoretical and the applicative side. However, in spite of their strong results in low level learning tasks, there is still a gap with models rooted in logic and probability, whenever one needs to express relations and express constraints amongst different entities. This paper describes how kernel-like models, inspired by the parsimony principle, can cope with highly structured and rich environments that are described by the unified notion of constraint. We formulate the learning as a constrained variational problem and prove that an approximate solution can be given by a kernel-based machine, referred to as a *support constraint machine* (SCM), that makes it possible to deal with learning tasks (functions) and constraints. The learning process resembles somehow the unification of Prolog, since the learned functions yield the verification of the given constraints. Experimental evidence is given of the capability of SCMs to check new constraints in the case of first-order logic.

Key words: Kernel machines, Learning from constraints, Support vector machines.

1 Introduction

This paper evolves a general framework of learning aimed at bridging logic and kernel machines [1]. We think of an intelligent agent acting in the perceptual space $\mathcal{X} \subset \mathbb{R}^d$ as a vectorial function $f = [f_1, \ldots, f_n]'$, where $\forall j \in \mathbb{N}_n : \ \ f_j \in W^{k,p}$ belongs to a Sobolev space, that is to the subset of L^p whose functions f_j admit weak derivatives up to some order k and have a finite L^p norm. The functions $f_j : \ \ j = 1, \ldots, n$, are referred to as the "tasks" of the agent. We can introduce a norm on f by the pair (P, γ), where P is a pseudo-differential operator and $\gamma \in \mathbb{R}^n$ is a vector of non-negative coordinates

$$R(f) = \parallel f \parallel^2_{P_\gamma} = \sum_{j=1}^{n} \gamma_j < P f_j, P f_j >, \tag{1}$$

which is used to determine smooth solutions according to the parsimony principle. This is a generalization to multi-task learning of what has been proposed in [2] for regularization networks. The more general perspective suggests considering objects as entities picked up in $\mathcal{X}^{p,\star} = \bigcup_{i \le p} \bigcup_{|\alpha_i| \le p^{\underline{i}}} \mathcal{X}_{\alpha_{1,i}} \times \mathcal{X}_{\alpha_{2,i}}, \ldots, \mathcal{X}_{\alpha_{i,i}}$ where $\alpha_i = \{\alpha_{1,i}, \ldots, \alpha_{i,i}\} \in \mathcal{P}(p,i)$ is any of the $p^{\underline{i}} = p(p-1)\ldots(p-i+1)$ (falling factorial power of p) i-length sequences without repetition of p elements. In this paper, however, we restrict the analysis to the case in which the objects are simply points of a vector space. We propose to build an interaction amongst different tasks by introducing constraints of the following types [1]

$$\forall x \in \mathcal{X}: \ \phi_i(x, y(x), f(x)) = 0, \ i \in I\!N_m$$

where $I\!N_m$ is the set of the first m integers, and $y(x) \in I\!R$ is a *target function*, which is typically defined only on samples of the probability distribution. This makes it possible to include the classic supervised learning, since pairs of labelled examples turns out to be constraints given on a finite set of points. Notice that one can always reduce a collection of constraints to a single equivalent constraint. For this reason, in the reminder of the paper, most of the analysis will focus on single constraints. In some cases the constraints can be profitably relaxed and the index to be minimized becomes

$$R(f) = \| f \|^2_{P_\gamma} + C \cdot 1' \int_{\mathcal{X}} \varXi(x, y(x), f(x)). \tag{2}$$

where $C > 0$ and the function $\varXi$ penalizes how we depart from the perfect fulfillment of the vector of constraints ϕ, and 1 is a vector of ones. If $\phi(x, y(x), f(x)) \ge 0$ then we can simply set $\varXi(x, y(x), f(x)) := \phi(x, y(x), f(x))$, but in general we need to set the penalty properly. For example, the check of a bilateral constraint can be carried out by posing $\varXi(x, y(x), f(x)) := \phi^2(x, y(x), f(x))$.

Of course, different constraints can represent the same admissible functional space $\mathcal{F}_\phi$. For example, constraints $\check{\phi}_1(f, y) = \epsilon - |y - f| \ge 0$ and $\check{\phi}_2(f, y) = \epsilon^2 - (y - f)^2 \ge 0$ where f is a real function, define the same $\mathcal{F}_\phi$. This motivates the following definition.

Definition 1. *Let $\mathcal{F}_{\phi_1}, \mathcal{F}_{\phi_2}$ be the admissible spaces of ϕ_1 and ϕ_2, respectively. Then we define the relation $\phi_1 \sim \phi_2$ if and only if $\mathcal{F}_{\phi_1} = \mathcal{F}_{\phi_2}$.*

This notion can be extended directly to pairs of collection of constraints, that is $\mathcal{C}_1 \sim \mathcal{C}_2$ whenever there exists a bijection $\mathcal{C}_1 \overset{\nu}{\to} \mathcal{C}_2$ such that $\forall \phi_1 \in \mathcal{C}_1 \ \ \nu(\phi_1) \sim \phi_1$. Of course, $\sim$ is an equivalent relation. We can immediately see that $\phi_1 \sim \phi_2 \Leftrightarrow \forall f \in \mathcal{F}: \ \exists P(f): \ \ \phi_1(f) = P(f) \cdot \phi_2(f)$, where P is any positive real function. Notice that if we denote by $[\phi]$ a generic representative of $\sim$, than the quotient set $\mathcal{F}_\phi / \sim$ can be constructed by

$$\mathcal{F}_\phi / \sim = \{\phi \in \mathcal{F}_\phi : \ \phi = P(f) \cdot [\phi](f)\}.$$

Of course we can generate infinite constraints equivalent to $[\phi]$. For example, if $[\phi(f, y) = \epsilon - |y - f|]$, the choice $P(f) = 1 + f^2$ gives rise to the equivalent

[1] We restrict the analysis to universally-quantified constraints, but a related analysis can be carried out when involving existential quantifiers.

constraint $\phi(f,y)=(1+f^2)\cdot(\epsilon-|y-f|)$. The quotient set of any single constraint ϕ_i suggests the presence of a *logic structure*, which makes it possible to devise reasoning mechanisms with the representative of the relation $\sim$. Moreover, the following notion of entailment naturally arises:

Definition 2. *Let $\mathcal{F}_{\overline{\phi}} = \left\{f \in \mathcal{F} : \ \overline{\phi}(f) \geq 0\right\}$. A constraint $\overline{\phi}$ is entailed by $\mathcal{C} = \{\phi_i, i \in I\!N_m\}$, that is $\mathcal{C} \models \overline{\phi}$, if $\mathcal{F}_{\mathcal{C}} \subset \mathcal{F}_{\overline{\phi}}$.*

Of course, for any constraint ϕ that can be formally deduced from the collection $\mathcal{C}$ (premises), we have $\mathcal{C} \models \overline{\phi}$. It is easy to see that the entailment operator states invariant conditions in the class of equivalent constraints, that is if $\mathcal{C} \sim \mathcal{C}'$, $\mathcal{C} \models \phi$, and $\phi \sim \phi'$ then $\mathcal{C}' \models \phi'$. The entailment operator also meets the classic chain rule, that is if $\mathcal{C}_1 \models \mathcal{C}_2$ and $\mathcal{C}_2 \models \mathcal{C}_3$ then $\mathcal{C}_1 \models \mathcal{C}_3$.

2 SCM for Constraint Checking

A dramatic simplification of the problem of learning from constraints derives from sampling the input space $\mathcal{X}$, so as to restrict their verification on the set $[\mathcal{X}]_\ell := \{\boldsymbol{x}_\kappa \in \mathcal{X}, \ \kappa \in I\!N_\ell\}$. This typically cannot guarantee that the algorithm will be able to satisfy the constraint over the whole input space. However, in this work we consider that there is a marginal distribution $\mathcal{P}_{\mathcal{X}}$ that underlies the data in $\mathcal{X}$, as it is popularly assumed by the most popular machine learning algorithms, so that the constraint satisfaction will holds with high probability.

Theorem 1. *Given a constraint ϕ, let us consider the problem of learning from*

$$\forall \kappa \in I\!N_\ell : \ \phi(x_\kappa, y(x_\kappa), f(x_\kappa)) = 0. \tag{3}$$

There exist a set of real constants $\lambda_\kappa, \quad \kappa \in I\!N_\ell$ such that any weak extreme of functional (1) that satisfies (3) is also a weak extreme of $E_\phi(f) = \parallel f \parallel^2_{P_\gamma} + \sum_{\kappa \in I\!N_\ell} \lambda_\kappa \cdot \phi(x_\kappa, f(x_\kappa))$. The extreme $f^\star$ becomes a minima if the constraints are convex, and necessarily satisfy the Euler-Lagrange equations $Lf^\star(x) + \sum_{\kappa=1}^{\ell} \lambda_\kappa \cdot \nabla_f \phi(x_\kappa, y(x_\kappa), f^\star(x_\kappa))\delta(x - x_\kappa) = 0$ being $L := P'P$, where P' is the adjoint of P. Moreover, let us assume that $\forall x \in \mathcal{X} : \ g(x, \cdot)$ be the Green function of L. The solution $f^\star$ admits the representation

$$f^\star(x) = \sum_{\kappa \in I\!N_\ell} a_k \cdot g(x, x_\kappa) + f_P(x), \tag{4}$$

where $a_\kappa = -\lambda_\kappa \nabla_f \phi(x_\kappa, y(x_\kappa), f(x_\kappa))$. The uniqueness of the solution arises, that is $f_P = 0$, whenever $KerP = \{0\}$. If we soften the constraint (3) then all the above results still hold when posing $\forall \kappa \in I\!N_\ell : \ \lambda_\kappa = C$.

PROOF: (Sketch)
Let $\mathcal{X} = I\!R^d$ be and let $\{\zeta_h(x)\}_{h=1}^{\infty}$ be a sequence of mollifiers and choose $\epsilon := 1/h$, where $h \in I\!N$. Then $\{\zeta_h(x)\}_{h=1}^{\infty} \overset{weakly}{\longrightarrow} \delta(x)$ converges in the classic weak limit sense to the delta distribution. Given the single constraint ϕ let us

consider the sampling $\mathcal{F}_\phi \overset{\mathcal{S}}{\to} \mathcal{F}_\phi : \phi \to [\phi]_h$ carried out by the mollifiers ζ_h on $[\mathcal{X}]_\ell$

$$[\phi]_h(x, y(x), f(x)) := \sum_{\kappa \in I\!N_\ell} \phi(x, y(x), f(x))\zeta_h(x - x_\kappa) = 0.$$

Of course, $[\phi]_h$ is still a constraint and, it turns out that as $h \to \infty$ it is equivalent to $\forall \kappa \in I\!N_\ell : \ \phi(x_\kappa, y(x_\kappa), f(x_\kappa)) = 0$. Now the proof follows by expressing the overall error index on the finite data sample for $[\phi]_h(x, y(x), f(x))$.

We can apply the classic Euler-Lagrange equations of variational calculus with subsidiary conditions for the case of holonomic constraints ([3] pag. 97-110), so as any weak extreme of (1) that satisfies (3) is a solution of the Euler-Lagrange equation

$$Lf^\star(x) + \sum_{\kappa \in I\!N_\ell} \lambda_i(x) \cdot \nabla_f \phi_i(x, y(x), f^\star(x))\zeta_h(x - x_\kappa) = 0, \tag{5}$$

where $L := [\gamma_1 L, \ldots, \gamma_n L]'$ and ∇_f is the gradient w.r.t. f. The convexity of the constraints guarantees that the extreme is a minima and $KerP = \{0\}$ ensures strict convexity and, therefore, uniqueness. Finally, (4) follows since $\forall x \in \mathcal{X} : g(x, \cdot)$ is the Green function of L. The case of soft-constraints can be treated by similar arguments. ■

From (4), which give the representation of the optimal solution we can collapse the dimensionality of $\mathcal{F}$ and search for solutions in a finite space. This is stated in the following theorem.

Theorem 2. *Let us consider the learning under the sampled constraints (3). In the case $kerP = \{0\}$ we have $f_P = 0$ and the optimization is reduced to the finite-dimensional problem*

$$\min_a \left\{ \sum_{\kappa \in I\!N_n} \gamma_j a_j' G a_j + \sum_{\kappa \in I\!N_\ell} \lambda_k \phi(x_\kappa, y(x_\kappa), f^\star(x_\kappa)) \right\} \tag{6}$$

that must hold jointly with (3). If $\phi \geq 0$ holds for an equality soft-constraint ϕ then the above condition still holds and, moreover, $\forall \kappa = 1, \ldots, \ell : \ \lambda_\kappa = C$.

Proof: The proof comes out straightforwardly when plugging the expression of $f^\star$ given by (4) into $R(f)$. ■

For a generic bilateral soft-constraint we need to construct a proper penalty. For example we can find $\arg\min_a \left\{ \sum_{j=1}^n \gamma_j a_j' G a_j + C \sum_{\kappa=1}^\ell \phi^2(x_\kappa, f(x_\kappa)) \right\}$. Then, the optimal coefficients a can be found by gradient descent, or using any other efficient algorithm for unconstrained optimization. In particular, we used an adaptive gradient descent to run the experiments. Note that when ϕ is not convex, we may end up in local minima.

Theorem 2 can be directly applied to classic formulation of learning from examples in which $n = 1$ and $\phi = \Xi$ is a classic penalty and yields the classic optimization of $\arg\min_a \{a'Ga + C\sum_{\kappa=1}^{\mathcal{S}_\ell} \Xi(x_\kappa, y(x_\kappa), f(x_\kappa))\}$. Our formulation of learning leads to discovering functions f compatible with a given collection of constraints that are as smooth as possible. Interestingly, the shift of focus on constraints opens the doors to the following *constraint checking problem*

Definition 3. *Let us consider the collection of constraints* $\mathcal{C} = \{\phi_i, i \in I\!N_m\} = \mathcal{C}_p \cup \mathcal{C}_c$ *where* $\mathcal{C}_p \cap \mathcal{C}_c = \emptyset$. *The* constraint checking problem *is the one of establishing whether or not* $\forall \phi_i \in \mathcal{C}_c \ \ \mathcal{C}_p \models \phi_i$ *holds true. Whenever we can find* $f \in \mathcal{F}$ *such that this happens, we say that* $\mathcal{C}_c$ *is entailed by* $\mathcal{C}_p$*, and use the notation* $\mathcal{C}_p \models \mathcal{C}_c$.

Of course, the entailment can be related to the quotient set $\mathcal{F}_\phi/\sim$ and its analysis in the space $\mathcal{F}_\phi$ can be restricted to the representative of the defined equivalent class. Constraint checking is somehow related to *model checking* in logic, since we are interested in checking the constraints $\mathcal{C}_s$, more than in exhibiting the steps which leads to the proof.

Now, let $\mathcal{C}_p \models \phi$ and $f^\star = \arg\min_{f\in\mathcal{F}_p} \parallel f \parallel_{P_\gamma}$. Then it is easy to see that $\phi(f^\star) = 0$. Of course, the vice versa does not hold true. That if $f^\star = \arg\min_{f\in\mathcal{F}_p} \parallel f \parallel_{P_\gamma}$ and $\phi(f^\star) = 0 :\ \ \mathcal{C}_p \not\models \phi$. For example, consider the case in which the premises are the following collection of supervised examples $\mathcal{S}_1 := \{(x_\kappa, y_k)\}_{\kappa=1}^{\ell}$ and $\mathcal{S}_2 := \{(x_\kappa, -y_k)\}_{\kappa=1}^{\ell}$ given on the two functions f_1, f_2. It is easy to see that we can think of $\mathcal{S}_1$ and $\mathcal{S}_2$ in terms of two correspondent constraints ϕ_1 and ϕ_2, so as we can set $\mathcal{C}_p := \{\phi_1, \phi_2\}$. Now, let us assume that $\phi(f^\star) = f_1^\star - f_2^\star = 0$. This holds true whenever $a_{\kappa,1} = -a_{\kappa,2}$ Of course, the deduction $\mathcal{C} \models \phi$ is false, since f can take any value in outside the condition forced on supervised examples[2]. This is quite instructive, since it indicates that even though the deduction is formally false, the generalization mechanism behind the discovering of $f^\star$ yields a sort of approximate deduction.

Definition 4. *Let* $f^\star = \arg\min_{f\in\mathcal{F}_p} \parallel f \parallel_{P_\gamma}$ *be and assume that* $\phi(f^\star) = 0$ *holds true. We say that* ϕ *is* formally checked *from* $\mathcal{C}_p$ *and use the notation* $\mathcal{C}_p \vdash \phi$.

Interestingly, the difference between $\mathcal{C}_p \models \phi$ and $\mathcal{C}_p \vdash \phi$ is rooted in the gap between deductive and inductive schemes. While $\models$ does require a sort of unification by checking the property $\phi(f) = 0$ for all $f \in \mathcal{F}_p$, the operator $\vdash$ comes from the computation of $f^\star$ that can be traced back to the parsimony principle. Whenever we discover that $\mathcal{C}_p \vdash \phi$, it means that either $\mathcal{C}_p \models \phi$ or $f^\star \in \mathcal{F}_p \bigcap \mathcal{F}_\phi \subset \mathcal{F}_p$, where $\subset$ holds in strict sense. Notice that if we use soft-optimization then the notion of *simplification* strongly emerges which leads to a decision process in which more complex constraints are sacrificed because of the preference of simple constraints. We can go beyond $\vdash$ by relaxing the need to check $\phi(f^\star) = 0$ thanks to the introduction of the following notion of *induction from constraints.*

[2] Notice that the analysis is based on the assumption of hard-constraints and that in case of soft-constraints, which is typical in supervised learning, the claim that the deduction is false is even reinforced.

Definition 5. *Let $\epsilon > 0$ be and $[\mathcal{X}]_u \subset \mathcal{X}$ be a sample of u unsupervised examples of $\mathcal{X}$. Given a set of* premises *$\mathcal{C}_p$ on $[\mathcal{X}]_u$ and let $\mathcal{F}_p^u$ be the correspondent set of admissible functions. Furthermore, let $f^\star = \arg\min_{f \in \mathcal{F}_p^u} \parallel f \parallel_{P_\gamma}$ be and denote by $[f^\star]_u$ its restriction to $[\mathcal{X}]_u$. Now assume that $\parallel \phi([f^\star]_u) \parallel < \epsilon$ holds true. Under these conditions we say that ϕ is induced from $\mathcal{C}_p$ via $[\mathcal{X}]_u$, and use the notation $(\mathcal{C}_p, [\mathcal{X}]_u) \vdash^\star \phi$.*

Notice that the adoption of special loss functions, like the classic hinge function, gives rise to support vectors, but also to *support constraints.* Given a collection of constraints (premises) $\mathcal{C}_p$, then ϕ is a support constraint for $\mathcal{C}_p$whenever $\mathcal{C}_p \not\vdash \phi$. When the opposite condition holds, we can either be in presence of a formal deduction $\mathcal{C}_p \models \phi$ or of the more general checking $\mathcal{C}_p \vdash \phi$ in the environment condition.

3 Checking First-Order Logic Constraints

We consider the semi-supervised learning problem (6) composed of a set of constraints that include information on labeled data and prior knowledge on the learning environment in the form of First-Order Logic (FOL) clauses. Firstly, we will show how to convert FOL clauses in real-valued functions. Secondly, using an artificial benchmark, we will include them in our learning framework to improve the quality of the classifier. Finally, we will investigate the constraint induction mechanism, showing that it allows us to formally check other constraints that were not involved in the training stage (Definitions 4 and 5).

First-Order Logic (FOL) formula can be associated with real-valued functions by classic *t-norms* (triangular norms [4]). A t-norm is function $T\ :\ [0,1]\times[0,1] \to \mathbb{R}$, that is commutative, associative, monotonic and that features the neutral element 1. For example, given two unary predicates $a_1(\boldsymbol{x})$ and $a_2(\boldsymbol{x})$, encoded by $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$, the product norm, which meets the above conditions on T-norms, operates as follows: $a_1(\boldsymbol{x}) \wedge a_2(\boldsymbol{x}) \longmapsto f_1(\boldsymbol{x}) \cdot f_2(\boldsymbol{x})$, $a_1(\boldsymbol{x}) \vee a_2(\boldsymbol{x}) \longmapsto 1 - (1 - f_1(\boldsymbol{x})) \cdot (1 - f_2(\boldsymbol{x}))$, $\neg a_1(\boldsymbol{x}) \longmapsto 1 - f_1(\boldsymbol{x})$, and $a_1(\boldsymbol{x}) \Rightarrow a_2(\boldsymbol{x}) \longmapsto 1 - f_1(\boldsymbol{x}) \cdot (1 - f_2(\boldsymbol{x}))$. Any formula can be expressed by the CNF (Conjunctive Normal Form) so as to transform it to a real-valued constraint step by step. In the experiment we focus on universally quantified ($\forall$) logic clauses, but the extension to cases in which the existential quantifier is involved is possible.

We consider a benchmark based on 1000 bi-dimensional points belonging to 4 (partially) overlapping classes. In particular, 250 points for each class were randomly generated with uniform distribution. The classes a_1, a_2, a_3, a_4 can be thought of the characteristic functions of the domains $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4$ defined as $\mathcal{D}_1 = \{(x_1, x_2) \in \mathbb{R}^2\ :\ x_1 \in (0,2)\ \wedge\ x_2 \in (0,1)\}$, $\mathcal{D}_2 = \{(x_1, x_2) \in \mathbb{R}^2\ :\ x_1 \in (1,3)\ \wedge\ x_2 \in (0,1)\}$, $\mathcal{D}_3 = \{(x_1, x_2) \in \mathbb{R}^2\ :\ x_1 \in (1,2)\ \wedge\ x_2 \in (0,2)\}$, and $\mathcal{D}_4 = \{(x_1, x_2) \in \mathbb{R}^2\ :\ (x_1 \in (1,2)\ \wedge\ x_2 \in (0,1)) \vee\ (x_1 \in (0,2)\ \wedge\ x_2 \in (1,2))\}$. Then the appropriate multi-class label was assigned to the collection of 1000 points by considering their coordinates (see Fig. 1). A multi-class label is a binary vector of p components where 1 marks the membership to the i-th category (for example, $[0,1,1,0]$ for a point of classes a_2 and a_3). Four binary classifiers were trained using the associated functions f_1, f_2, f_3, f_4. The decision of each classifier on an input $\boldsymbol{x}$ is $o_j(\boldsymbol{x}) = \mathbf{1}(f_j(\boldsymbol{x}) - b_j)$ where b_j is the bias term of the j-th classifier and $\mathbf{1}(\cdot)$ is the Heaviside function.

We simulate a scenario in which we have access to the whole data collection, where ℓ points ($\ell/4$ for each class) are labeled, and to domain knowledge expressed by the following FOL clauses,

$$\forall \boldsymbol{x} \quad a_1(\boldsymbol{x}) \ \wedge a_2(\boldsymbol{x}) \Rightarrow a_3(\boldsymbol{x}) \tag{7}$$

$$\forall \boldsymbol{x} \quad a_3(\boldsymbol{x}) \Rightarrow a_4(\boldsymbol{x}) \tag{8}$$

$$\forall \boldsymbol{x} \quad a_1(\boldsymbol{x}) \ \vee a_2(\boldsymbol{x}) \ \vee \ a_3(\boldsymbol{x}) \ \vee \ a_4(\boldsymbol{x}). \tag{9}$$

While the first two clauses express relationships among the classes, the last clause specifies that a sample must belong to at least one class. For each of the ℓ labeled training points, we assume to have access to a *partial labeling*, such as, for example, $[0, ?, 1, ?]$, that means that we do not have any information on classes 2 and 4. This setup emphasizes the role of the FOL clauses in the learning process. We performed a 10-fold cross-validation and measured the average classification accuracy on the out-of-sample test sets. A small set of partially labeled data is excluded from the training splits, and it is only used to validate the classifier parameters (that were moved over $[0.1, 0.5, \ldots, 12]$ for the width of a Gaussian kernel, and $[10^{-4}, 10^{-3}, \ldots, 10^{2}]$ for the regularization parameter λ of soft-constrained SCM). We compared SCMs that include constraints on labeled points only (SCL$_L$) with SCMs that also embed constraints from the FOL clauses (that we indicate with SCM$_{FOL}$). In Fig. 1 we report a visual comparison of the two algorithms, where the outputs of f_1, f_2, f_3, f_4 are plotted ($\ell = 16$).

The introduction of the FOL clauses establish a relationship among the different classification functions and positively enhance the inductive transfer among the four tasks. As a matter of fact, the output of SCM$_{FOL}$ is significantly closer to the real class boundaries (green dashed lines) than in SCM$_L$. The missing label information is compensated by the FOL rules, and injected on the whole data distribution by the proposed learning from constraint scheme. Note that the missing labels cannot be compensated by simply applying the FOL clauses on the partially labeled vectors. Interestingly the classifier has learned the “shape” of the lower portion of class 3 by the rule of (7), whereas the same region on class 4 has been learned thanks to the inductive transfer from (8).

We iteratively increased the number of labeled training points ℓ from 8 to 320, and we measured the classification macro accuracies. The output vector on a test point $\boldsymbol{x}$ (i.e. $[o_1(\boldsymbol{x}), o_2(\boldsymbol{x}), o_3(\boldsymbol{x}), o_4(\boldsymbol{x})]$) is considered correctly predicted if it matches the full label vector that comes with the ground truth. We also computed the accuracy of the SCM$_L$ classifier whose output is post-processed by applying the FOL rules, in order to fix every incoherent prediction. The results are reported in Fig. 2. The impact of the FOL clauses in the classification accuracy is appreciable when the number of labeled points is small, whereas it is less evident when the information on the FOL clauses can be learned from the available training labels, as expected. Clearly the results becomes less significant when the information on the FOL clauses can be learned from the available

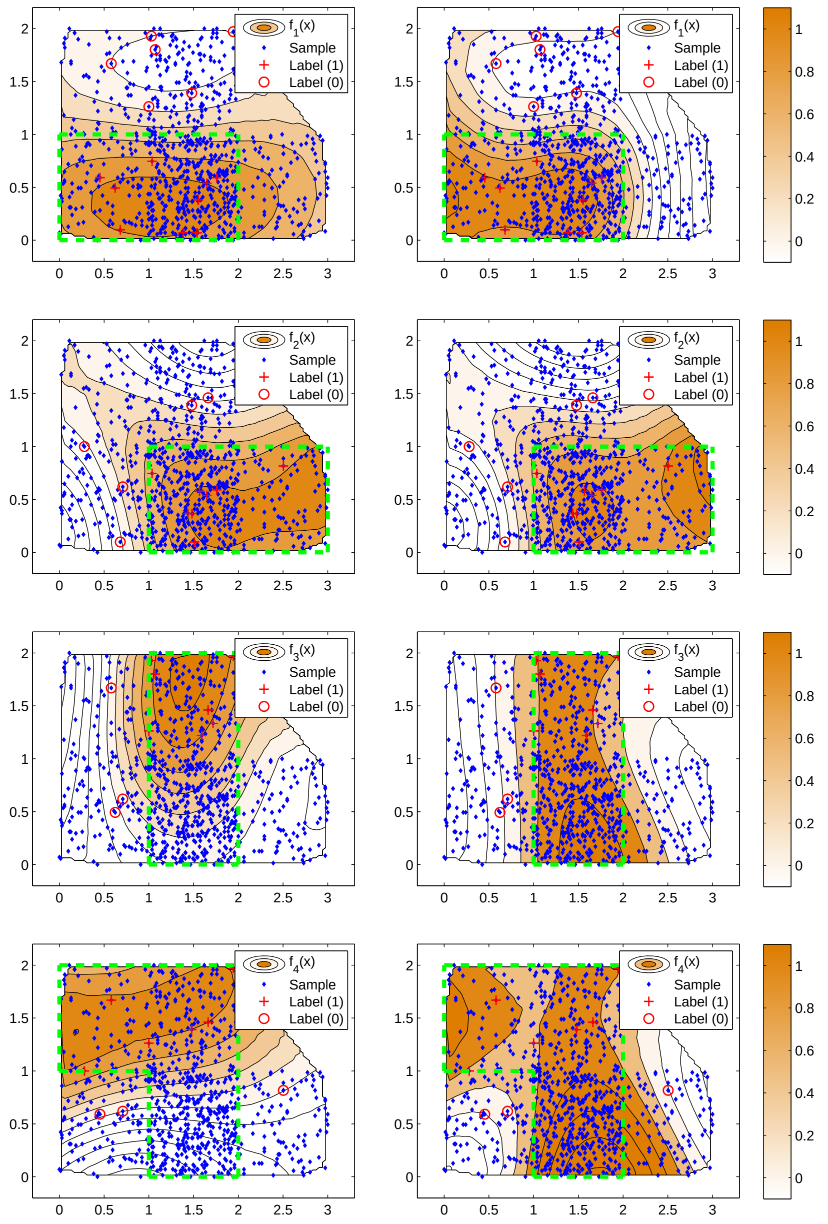

Fig. 1. The functions f_1,f_2,f_3,f_4 in the data collection where a set of FOL constraints applies. The j-th row, $j = 1, \ldots, 4$, shows the outcome of the function f_j in SCMs that use labeled examples only (left) and in SCMs with FOL clauses, SCM$_{FOL}$ (right). The green dashed lines shows the real boundaries of the j-th class.

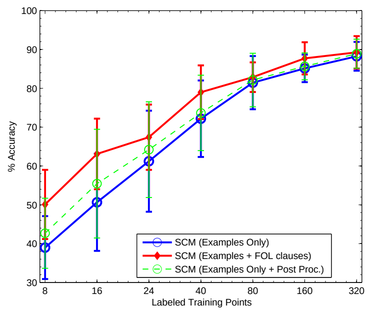

Fig. 2. The average accuracy (and standard deviation) of the SCM classifier: using labeled examples only (SCL$_L$), using examples and FOL clauses (SCM$_{FOL}$), using examples only and post processing the classifier output with the FOL rules.

training labels, so that in the rightmost region of the graph the gap between the curves becomes smaller. In general, the output of the classifier is also more stable when exploiting the FOL rules, showing a averagely smaller standard deviation.

Given the original set of rules that constitutes our Knowledge Base (KB) and that are fed to the classifier, we distinguish between two categories of logic rules that can be deducted from the trained SCM$_{FOL}$. The first category includes the clauses that are related to the geometry of the data distribution, and that, in other words, are strictly connected to the topology of the environment in which the agent operates, as the ones of (7-9). The second category contains the rules that can be logically deducted by analyzing the FOL clauses that are available at hand. The classifier should be able to learn both the categories of rules even if not explicitly added to the knowledge base.

The mixed interaction of the labeled points and the FOL clauses of the KB leads to an SCM agent that can check whether a *new* clause holds true in our environment. Note that the checking process is not implemented with a strict decision on the truth value of a logic sentence (*holds true or false*), since there are some rules that are verified only on some (possibly large) regions of the input space, so that we have to evaluate the *truth degree* of a FOL clause. If it is over a reasonably high threshold, the FOL sentence can be assumed to hold true. In Table 1 we report the degree of satisfaction of different FOL clauses and the Mean Absolute Error (MAE) on the corresponding t-norm-based constraints. We used the SCM$_{FOL}$ trained with $\ell = 40$. Even if it is simple to devise them when looking at the data distribution, it is not possible to do this as the input space dimension increases, so that we can only "ask" the trained SCM if a FOL clause holds true. This allow us to rebuild the hierarchical structure of the data, if any, and to extract compact information from the problem at hand. The rules belonging to the KB are accurately learned by the SCM$_{FOL}$, as expected. The SCM$_{FOL}$ is also able to deduct all the other rules that are supported in the entire data collection. The ones that do not hold for all the data points have the same truth degree as the percentage of points for which they should hold true,

Table 1. Mean Absolute Error (MAE) of the t-norm based constraints and the percentage of points for which a clause is marked *true* by the SCM (Average Truth Value), and their standard deviations (in brackets). Logic rules belong to different categories (Knowledge Base - KB, Environment - ENV, Logic Deduction - LD). The percentage of Support indicates the fraction of the data on which the clause holds true.

FOL clause	**Category**	**Support**	**MAE**	**Truth Value**
$a_1(\boldsymbol{x}) \wedge a_2(\boldsymbol{x}) \Rightarrow a_3(\boldsymbol{x})$	KB	100%	0.0011 (0.00005)	98.26% (1.778)
$a_3(\boldsymbol{x}) \Rightarrow a_4(\boldsymbol{x})$	KB	100%	0.0046 (0.0014)	98.11% (2.11)
$a_1(\boldsymbol{x}) \vee a_2(\boldsymbol{x}) \vee a_3(\boldsymbol{x})$	KB	100%	0.0049 (0.002)	96.2% (3.34)
$a_1(\boldsymbol{x}) \wedge a_2(\boldsymbol{x}) \Rightarrow a_4(\boldsymbol{x})$	LD	100%	0.0025 (0.0015)	96.48% (3.76)
$a_1(\boldsymbol{x}) \wedge a_3(\boldsymbol{x}) \Rightarrow a_2(\boldsymbol{x})$	ENV	100%	0.017 (0.0036)	91.32% (5.67)
$a_3(\boldsymbol{x}) \wedge a_2(\boldsymbol{x}) \Rightarrow a_1(\boldsymbol{x})$	ENV	100%	0.024 (0.014)	91.7% (4.57)
$a_1(\boldsymbol{x}) \wedge a_3(\boldsymbol{x}) \Rightarrow a_4(\boldsymbol{x})$	ENV	100%	0.0027 (0.0014)	96.13% (3.51)
$a_2(\boldsymbol{x}) \wedge a_3(\boldsymbol{x}) \Rightarrow a_4(\boldsymbol{x})$	ENV	100%	0.0025 (0.0011)	96.58% (4.13)
$a_1(\boldsymbol{x}) \wedge a_4(\boldsymbol{x})$	ENV	46%	0.41 (0.042)	45.26% (5.2)
$a_2(\boldsymbol{x}) \vee a_3(\boldsymbol{x})$	ENV	80%	3.39 (0.088)	78.26% (6.13)
$a_1(\boldsymbol{x}) \vee a_2(\boldsymbol{x}) \Rightarrow a_3(\boldsymbol{x})$	ENV	65%	0.441 (0.0373)	68.28% (5.86)
$a_1(\boldsymbol{x}) \wedge a_2(\boldsymbol{x}) \Rightarrow \neg a_4(\boldsymbol{x})$	LD	0%	0.26 (0.06)	3.51% (3.76)
$a_1(\boldsymbol{x}) \wedge \neg a_2(\boldsymbol{x}) \Rightarrow a_3(\boldsymbol{x})$	ENV	0%	0.063 (0.026)	27.74% (18.96)
$a_2(\boldsymbol{x}) \wedge \neg a_3(\boldsymbol{x}) \Rightarrow a_1(\boldsymbol{x})$	ENV	0%	0.073 (0.014)	5.71% (5.76)

whereas rules that do not apply to the given problem are correctly marked with a significantly low truth value.

4 Conclusions

This paper gives insights on how to fill the gap between kernel machines and models rooted in logic and probability, whenever one needs to express relations and express constraints amongst different entities. The *support constraint machines* (SCMs), are introduced that makes it possible to deal with learning functions in a multi-task environment and to check constraints. In addition to the impact in multi-task problems, the experimental results provide evidence of novel inference mechanisms that nicely bridge formal logic reasoning with supervised data. It is shown that logic deductions that do not hold formally can be fired by samples of labeled data. Basically SCMs provide a natural mechanism under which logic and data complement each other.

References

1. Diligenti, M., Gori, M., Maggini, M., Rigutini, L.: Bridging logic and kernel machines. Machine Learning (2011) to appear, on–line May 2011
2. Poggio, T., Girosi, F.: A theory of networks for approximation and learning. Technical report, MIT (1989)
3. Giaquinta, M., Hildebrand, S.: Calculus of Variations I. Volume 1. Springer (1996)
4. Klement, E., Mesiar, R., Pap, E.: Triangular Norms. Kluwer Academic Publisher (2000)